

# Pixel-wise video stabilization

Zhongqiang Wang<sup>1</sup> · Hua Huang<sup>2</sup>

Received: 8 August 2014 / Revised: 25 June 2015 / Accepted: 24 August 2015 /  
Published online: 21 September 2015  
© Springer Science+Business Media New York 2015

**Abstract** In this paper, we present a novel video stabilization method with a pixel-wise motion model. In order to avoid distortion introduced by traditional feature points based motion models, we focus on constructing a more accurate model to capture the motion in videos. By taking advantage of dense optical flow, we can obtain the dense motion field between adjacent frames and set up a pixel-wise motion model which is accurate enough. Our method first estimates dense motion field between adjacent frames. A PatchMatch based dense motion field estimation algorithm is proposed. This algorithm is specially designed for similar video frames rather than arbitrary images to reach higher speed and better performance. Then, a simple and fast smoothing algorithm is performed to make the jittered motion stabilized. After that, we warp input frames using a weighted average algorithm to construct the output frames. Some pixels in output frames may be still empty after the warping step, so in the last step, these empty pixels are filled using a patch based image completion algorithm. We test our method on many challenging videos and demonstrate the accuracy of our model and the effectiveness of our method.

**Keywords** Pixel-wise video stabilization · Dense motion field · Motion smoothing · Image completion

---

✉ Hua Huang  
huahuang@bit.edu.cn

Zhongqiang Wang  
wangzhongqiang@stu.xjtu.edu.cn

<sup>1</sup> School of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an, China

<sup>2</sup> Beijing Key Lab of Intelligent Information Technology, School of Computer Science, Beijing Institute of Technology, Beijing, China

## 1 Introduction

For amateurs, hand-held video camera is an easy-to-use tool to shoot videos. It can be conveniently used by people without any knowledge about photography, even a child. Therefore, it is getting more and more popular in recent years. Meanwhile, the quality of videos captured with hand-held cameras is still much lower than the quality of videos captured using professional devices. The main problem for videos captured by hand-held cameras is jitter. Professionals use special equipments, such as dolly tracks and steadicams, to make the path of camera smooth and under control. However, camera path of hand-held camera cannot be well controlled because of shake of hands and movement of photographer. Jitter decreases the quality of videos very much. Therefore, the technique of stabilizing a jittered video is necessary, which is called *video stabilization*.

Morimoto et al. [21] abstract video stabilization into three steps: motion estimation, motion compensation and image reconstruction. Prior video stabilization methods can be generally divided into two categories based on the motion model they use in the first two steps. The first class is 2D methods. In these methods, every frame is considered as a single plane and a 2D transformation (usually affine or perspective) between two frames is used to model the motion. The video is stabilized by smoothing the parameters of these transformation matrices. These methods are generally computationally efficient because the models they use are very simple. But on the contrary, these simple models cannot handle videos with complex scenes well, such as scenes containing multiple planes or large depth change. The second class is 3D methods. These methods recover the full 3D structure of the scenes in the jittered videos. After that, videos can be easily stabilized by directly smoothing the camera path or simply designing the camera path. 3D methods can stabilize videos with complex scenes on which 2D methods may fail. However, this class of methods highly depends on Structure from Motion (SfM) method to recover the 3D structure of the scenes. Since SfM methods are usually very slow and not robust enough, 3D stabilization methods cannot work on videos without enough 3D structure information and usually very slow.

Most previous methods set up a motion model depending on feature points correspondences (trajectories). However, in the image reconstruction step, every pixel in the output video frames, rather than only several feature points, should be assigned a value. In previous methods, homographies are wildly used to solve this problem. Many works use single homography warping to generate output frames. This is a 2D assumption which suffers from the disadvantages of all 2D methods, i.e. it is not capable to model complex scenes. In order to improve the performance of single homography warping, Liu et al. [13] propose a new warping method named content preserving warping (CPW). This is a warping method using multiple homographies. It splits each frame into many grids and warps each grid using a different homography. While being robust enough, this technique can generate more plausible results than single homography warping. CPW makes an improvement in the image reconstruction step but its motion estimation step and motion compensation step are still based on feature points. Liu et al. [16] propose a method which introduces mesh-based idea into motion estimation step. This method splits every input frame into grids in the first place and then models camera motion with a bundle of camera paths, each of which belongs to a grid cell. An optimization based smoothing algorithm is performed on this bundle of camera paths and then the multiple warping method of CPW can be combined with this method seamlessly. Compared to other works, this method greatly reduces the use of feature points. Feature points are only used when extracting the motion of grids between adjacent frames and feature points are matched only between adjacent frames.

From the above we can see that recent methods are trying to avoid the disadvantage of using feature points. [13] and [16] try to handle all pixels in the frames rather than only the detected feature points. In fact, a feature points guided homography cannot represent the motion of a whole frame well because the number of feature points are usually much less than the number of all pixels in one frame. Homographies are used to warp all pixels in each frame, but they are calculated only using a few feature points. Therefore, the performance is not uniform in frames. It is better in area with more feature points but worse in area with less feature points.

In this paper, we aim to stabilize jittered videos using a motion model which can represent the motion between frames more precisely than previous models based on feature points. To this end, we propose a *pixel-wise* motion model which calculates the dense motion field for every pair of adjacent frames. This pixel-wise motion model is based on dense optical flow using nearest neighbour fields [1]. The corresponding position of one pixel in another frame is calculated by finding the nearest neighbour patch. With the help of dense optical flow, correspondences of all pixels between adjacent frames can be found and thus the motion model is dense and uniform. Every pixel takes part in the stabilization procedure, so their stabilized positions do not need to be approximated by other pixels (warping using homographies). Therefore, this dense motion model is more precise than models based on sparse feature points. In addition to the new motion model, we also propose a simple and fast smoothing algorithm, a pixel-wise frame warping algorithm and a patch based image completion algorithm to fulfill the stabilization.

The main novelty of our work is that it stabilizes videos in a pixel-wise manner in the whole stabilization process. [16] handles videos by a regular grid mesh, which is the most similar method to ours so far. In fact, our method can be considered as the extreme situation of [16], i.e., the grid size is only one pixel. But the algorithm of our method is totally different from [16]. The contribution of our method lies in the following aspects:

1. A novel dense motion estimation algorithm is proposed. Correspondence for every pixel is searched from adjacent frames. Since there is strong spacial coherency between adjacent frames, the searching range is greatly narrowed down by taking advantage of it.
2. A pixel-wise frame warping algorithm is proposed. We use a weighted average strategy to make most pixels in output frames get correct values. Some empty pixels may still exist after warping. Finally, a simple patch based image completion algorithm is proposed to fill them.

## 2 Related works

### 2.1 Video stabilization methods

According to the dimensionality of the motion model they use, previous video stabilization methods can be divided into two categories: 2D methods and 3D methods.

#### 2.1.1 2D methods

The basic assumption of 2D methods is that all the scenes are on the same plane. Under this assumption, all pixels in one frame can be handled by one single transformation matrix.

Some early methods such as [20] use a four-parameter rigid transformation and some methods such as [9] use a six-parameter affine transformation. Later 2D methods such as [5, 19] always use an eight-parameter perspective transformation because it is more flexible. Litvin et al. [12] construct a chain of affine transformations among continuous frames. They treat jitter as noise in the chain and propose a probabilistic model to filter the noise. This method suffers from cumulative error obviously. With the increase of frame number, the error becomes larger and larger. In order to avoid cumulative error, Matsushita et al. [19] propose a local method that filters the motion of each frame only within a few neighbour frames. Gleicher et al. [5] speculate users' intention in operating the camera and carefully design the transformations to fulfill the intentional motion rather than only simply filtering the transformations. It assumes the video segments to be either static or moving and calculates the transformation matrices using mosaic-based technique. This method can generate film-like effect. Similar with [5], Grundmann et al. [7] classify camera motion into the following three categories: constant, linear and parabolic, depending on the high order derivatives of trajectories. Then trajectories are smoothed using  $L_1$ -optimization. Rather than compensating inter-frame transformations directly, Lee et al. [11] seek the transformations that can make the trajectories as smooth as possible. Walha et al. [25] present a video stabilization and moving object detection system based on camera motion estimation. Most methods use SIFT [17] or KLT [23] to find and track feature points. Instead, Okade et al. [22] use Maximally Stable Extremal Region (MSER) which is fast and robust for video stabilization purpose.

Many recent methods, which also belong to 2D methods, directly calculate the stabilized position of feature points. In order to avoid distortion, they take some different constraints into consideration. Liu et al. [14] propose a video stabilization method based on an observation that smoothed trajectories should approximately lie in a low-rank subspace. They enforce subspace constraints on the trajectories matrix to get the smoothed trajectories. In order to avoid the reconstruction of 3D structure, Goldstein et al. [6] present a method to smooth trajectories utilizing the constraint of epipolar geometry. Wang et al. [26] filter trajectories while keeping similarity constraints among neighbour trajectories using triangulation. Wang et al. [27] detect multiplane structure from videos and then stabilize every plane independently. Liu et al. [16] make an improvement upon [13]. In this method, frames are divided into grids immediately after trajectory extraction step and the motion of grids are calculated using the motion of trajectories. Not only the warping step but also the stabilization step is applied using multiple homographies.

2D methods are usually fast and robust, but they cannot handle videos with complex scenes because of the single plane assumption. If the scene is constructed by more than one plane, the motion between frames cannot be modelled using only one homography. In contrast, the motion model of our method is much more flexible. Since we deal with every pixel separately, the motion between two frames is modelled using dense optical flow. Therefore, no matter how complex the scene is, our method can handle it well.

### 2.1.2 3D methods

3D methods recover the full 3D structure of the scene and the camera motion. Videos can be stabilized easily by smoothing or planning the path of camera directly. Zhang et al. propose a 3D video stabilization method [28]. It requires intrinsic and extrinsic parameters of the

camera of every frame as input of stabilization step and calculates the stabilized camera parameters using optimization. Liu et al. [13] propose a new warping method in order to generate perceptual plausible output frames. This method divides every frame into grids and applies spatially-varying warps according to the smoothed trajectories. Zhou et al. propose another 3D method [29]. It detects planes from videos and improves [13] to achieve higher performance on videos with multiple planes. Chouvatut et al. [3] estimate camera motion and reconstruct human face from a video sequence.

If the dense 3D structure of the scenes can be recovered, 3D methods can generate results easily. However, it is too difficult to recover dense 3D structure from videos. In fact, nearly all previous 3D methods recover only sparse 3D structure, except that Liu et al. [15] use a depth camera to get additional dense depth information. Dense optical flow is also used in [19], but it is only used to find the 2D transformation matrix between two frames. Recently, Liu et al. [24] propose a method which also stabilize videos in a pixel-wise manner. Instead of smoothing trajectories, this method smooths pixel profile, which collects motion vectors at the same pixel location, to stabilize jittered videos.

3D methods can generate many satisfied results, but they are usually not robust enough. They strongly depend on Structure from Motion (SfM) methods, which are used to recover the 3D structure of the video. However, even state-of-art SfM methods may fail in some situation, leading to failure cases in 3D video stabilization methods. On the contrary, our method depends on dense optical flow, which seldom fails. Therefore, our method is more robust than 3D methods.

## 2.2 PatchMatch based dense optical flow

Optical flow is a basic and important technique in computer vision which has been studied for a very long time. Sparse optical flow methods extract and track feature points in videos, while dense optical flow methods do that for every pixel. In sparse methods, only highly reliable points are chosen as feature points, so the number of feature points are much less than the number of all pixels. There is a large number of works on optical flow. Here we only discuss some of them which are strongly related to our work.

PatchMatch [1] is an algorithm to match similar patches between a pair of images. For one patch in one image, it can find the most matched patch in another image by calculating *Nearest-Neighbour Field* (NNF). Instead of using a naive brute force searching algorithm, [1] proposes a fast searching algorithm. This algorithm is based on an observation that since the image is continuous, the displacements of best matches of neighbour patches should also be in the neighbour. Block matching is also used for motion estimation. Block based motion estimation can also be used. Some fast techniques are proposed in [4].

PatchMatch [1] is speeded up by many following methods. Korman et al. [10] use hashing to make information propagate faster. He et al. [8] use Propagation-Assisted KD-Trees to avoid the time-consuming backtracking in traditional tree method. PatchMatch [1] is further used to compute dense optical flow. Chen et al. [2] use NNF of PatchMatch to compute an initial motion field and refine it as a motion segmentation problem. PatchMatch Filter [18] combines PatchMatch-based randomized search and efficient edge-aware image filtering together to get a state-of-the-art dense optical flow algorithm with high accuracy and speed.

### 3 Our approach

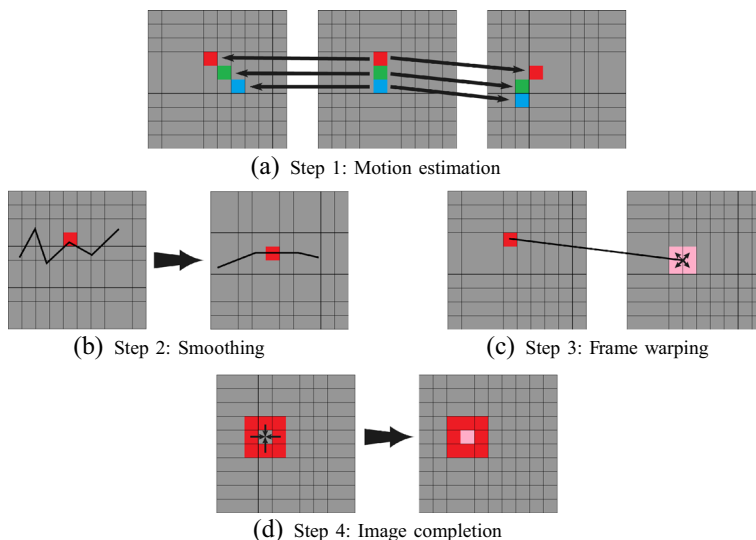
#### 3.1 Overview

Given a jittered input video which consists of  $n$  frames, we denote the input frames as  $\{I_i\}$ ,  $i \in [1, n]$ . We assume that the size of the frames is  $h$  by  $w$ , then every pixel can be denoted by its coordinate  $\mathbf{x} = (x, y)$ ,  $x \in [1, h]$ ,  $y \in [1, w]$ . The output of our method is a stabilized video which consists of  $n$  frames denoted as  $\{O_i\}$ ,  $i \in [1, n]$ .

The pipeline of our method is shown in Fig. 1. We can see that our method mainly consists of four steps. The first step is motion estimation step. For every frame in the input video, we detect the motion of every pixel between this frame and each adjacent frame to form a dense motion field. For every pixel  $\mathbf{x}$  in frame  $I_i$ , we find the corresponding pixel in frame  $I_{i-1}$  and denote the position of the corresponding pixel as  $C_i^{i-1}(\mathbf{x})$ . Similarly, we also find the corresponding pixel  $C_i^{i+1}(\mathbf{x})$  in frame  $I_{i+1}$ . In order to achieve high performance and efficiency, we propose a PatchMatch based dense optical flow algorithm which focuses on adjacent video frames with strong spacial coherency.

The second step is motion smoothing step. In this step, we calculate the stabilized position  $S_i(\mathbf{x})$  for every pixel  $\mathbf{x}$  in input frame  $I_i$ . Similar to smoothing a trajectory, low-pass filters can be used. In our experiments, Gaussian filter is used for simplicity. Although we only know the motion between adjacent frames, we can calculate the motion between every two frames by joining the motion of every two adjacent frames between these two frames.

The third step is frame warping step. After the two steps above, we have known the stabilized position of every pixel in input frames. In order to construct the output frames, we need to warp  $I_i$  to  $O_i$  using these correspondences. If the stabilized positions are all integers, this step can be easily done by directly copying the values. But since the original integer coordinates are smoothed in the second step, the stabilized positions may not be integers. So we use an weighted average algorithm inspired by bilinear interpolation to calculate an average value for every pixel in output frames.



**Fig. 1** Algorithm pipeline

The last step is image completion step. It cannot be guaranteed that all the pixels in output frames are assigned a value in the third step, so we propose an image completion algorithm to fill the empty pixels. Since the number of empty pixels are very small compared to the number of all pixels, we can calculate the corresponding position in input frame for every empty pixel using a patch based algorithm. Corresponding pixel’s value is copied to fill the empty pixels.

### 3.2 Motion estimation

In this section, we describe the dense optical flow based motion estimation algorithm in detail. For every pixel  $\mathbf{x}$  in every input frame  $I_i$ , the goal of our motion estimation algorithm is to find the corresponding positions in  $I_{i-1}$  and  $I_{i+1}$ :

$$\begin{aligned} \mathbf{x} &\rightarrow C_i^{i-1}(\mathbf{x}) \\ \mathbf{x} &\rightarrow C_i^{i+1}(\mathbf{x}) \end{aligned} \tag{1}$$

Denote the patch centering at  $\mathbf{x}$  in  $I_i$  as  $P_i(\mathbf{x})$ . For every pixel  $\mathbf{x}$  in  $I_i$ , we get the corresponding pixel in  $I_{i-1}$  by searching for the nearest neighbour patch for the patch  $P_i(\mathbf{x})$ :

$$C_i^{i-1}(\mathbf{x}) = \arg \min_{\mathbf{y} \in I_{i-1}} \|P_i(\mathbf{x}) - P_{i-1}(\mathbf{y})\| \tag{2}$$

$\|\cdot\|$  denotes the  $L_2$  norm. Obviously, it is not practical to do this search exhaustively. Since most area of an image is continuous, spacial coherency constraint should be hold. Equation (2) is improved to:

$$C_i^{i-1}(\mathbf{x}) = \arg \min_{\mathbf{y} \in I_{i-1}} (\|P_i(\mathbf{x}) - P_{i-1}(\mathbf{y})\| + w \cdot \sum_{\mathbf{x}' \in N(\mathbf{x})} \|C_i^{i-1}(\mathbf{x}') - \mathbf{y}\|) \tag{3}$$

where  $N(\mathbf{x})$  contains the four neighbour pixels of  $\mathbf{x}$  and  $w$  is a weight to balance the data term and smoothing term.

It is very difficult to solve (3) directly. PatchMatch [1] uses an iterative strategy. In every iteration, it searches two patches according to its two neighbour pixels and a random patch. Result is updated by the best one. This strategy greatly reduces computation, but since PatchMatch is designed for two arbitrary images, the searching range is still too wide for adjacent frames.

We still take advantage of the iterative strategy of PatchMatch. Moreover, we greatly narrow down the searching range for every pixel based on the following observation: it can be intuitively observed that, if two pixels are neighbour in one frame, their corresponding pixels in the other frame will be very near, usually within several pixels. Since two adjacent frames are captured almost at the same time under the same scene (except for sudden scene change, which is generally not considered in video stabilization methods), the motion between these two frames approximates to translation. It is obviously that two neighbour pixels are still in neighbour after a translation motion.

Now we explain our motion estimation algorithm in detail. Assuming we are dealing with frames  $I_i$  and  $I_{i-1}$ .

1. At first, we initialize the dense motion field  $C_i^{i-1}(\mathbf{x})$  with random value and run original PatchMatch for 1-2 iterations. After that, a large number of pixels in  $I_i$  are correctly matched to  $I_{i-1}$ . Though not all pixels are matched correctly, the overall motion can be detected, which is used to further refine the motion field.

2. Next, we calculate the average motion for every pixel using neighbour pixels:

$$\bar{C}_i^{i-1}(\mathbf{x}) = \frac{\sum_{\mathbf{y} \in N(\mathbf{x})} C_i^{i-1}(\mathbf{y})}{|N(\mathbf{x})|} \tag{4}$$

If  $\|\bar{C}_i^{i-1}(\mathbf{x}) - C_i^{i-1}(\mathbf{x})\| > th$ , we use  $\bar{C}_i^{i-1}(\mathbf{x})$  to replace  $C_i^{i-1}(\mathbf{x})$ .  $|N(\mathbf{x})|$  denotes the element number of  $N(\mathbf{x})$  and  $th$  is set to 5 in our experiments.

3. Then we perform a modified version of PatchMatch. In this process, the window of random search is limited to a small window around  $C_i(\mathbf{x})$  ( $30 \times 30$  in our experiments) and we use the energy term in (3) instead of (2) to calculate distance between two patches.
4. Repeat step 2 and 3 for 3-5 times to get the final dense motion field.

### 3.3 Smoothing

When the motions of all pixels are known, the smoothing step is carried out to get the stabilized position of pixels. In fact, any low-pass filter can be used and we use simple Gaussian filter in our experiments.

According to Section 3.2, for every pixel at  $\mathbf{x}$  in frame  $I_i$ , we know the corresponding pixel in  $I_{i+1}$  is at  $C_i^{i+1}(\mathbf{x})$ . Since we also know that the corresponding pixel for  $\mathbf{x}'$  in  $I_{i+1}$  is at  $C_{i+1}^{i+2}(\mathbf{x}')$  in  $I_{i+2}$ , we can calculate the coordinate of corresponding pixel for pixel at  $\mathbf{x}$  in frame  $I_i$  as  $C_{i+1}^{i+2}(C_i^{i+1}(\mathbf{x}))$  in frame  $I_{i+2}$ . Generally we can propagate the motion to get the correspondence of this pixel in any frame. For example, the coordinate of corresponding pixel in frame  $I_j$  ( $j > i$ ) is:

$$C_i^j(\mathbf{x}) = C_{j-1}^j(C_{j-2}^{j-1}(\dots C_{i+1}^{i+2}(C_i^{i+1}(\mathbf{x})) \dots)) \tag{5}$$

For correspondence in frame  $I_k$  ( $k < i$ ), the equation is similar:

$$C_i^k(\mathbf{x}) = C_{k+1}^k(C_{k+2}^{k+1}(\dots C_{i-1}^{i-2}(C_i^{i-1}(\mathbf{x})) \dots)) \tag{6}$$

For every pixel  $\mathbf{x}$  in frame  $I_i$ , we calculate its stabilized position using Gaussian filter as follow:

$$S_i(\mathbf{x}) = \sum_{j \in W(i)} C_i^j(\mathbf{x}) \cdot g(j - i) \tag{7}$$

where  $W(i) = \{j | i - n \leq j \leq i + n\}$  is the indices of neighbour frames of frame  $I_i$  and  $g(k) = \frac{1}{\sqrt{2\pi}\sigma} e^{-k^2/2\sigma^2}$  is the Gaussian kernel.  $n$  controls the size of neighbour frames. A larger value of  $n$  leads to more smoothed results but lower speed. We find that  $n = 15$  is enough for most videos.  $\sigma$  controls the strength of smoothing and we set it to 50.

Some previous methods, such as [14, 26] smooth trajectories while keep some constraint between trajectories. However, we smooth every pixel independently without any constraint and get quite good result (please refer to Section 4 and accompanying video). This is attributed to the patch based dense motion estimation algorithm in Section 3.2. In fact, the relationships between neighbour pixels have been constrained in the motion estimation step by using patches. Similar to the situation between two adjacent frames in Section 3.2, the motion between  $I_i$  and  $O_i$  are nearly translation, so neighbour pixels are very likely to be neighbour after stabilization. Therefore, the distortion in output frames is too small to be observed.



### 3.4 Frame warping

Now we want to generate every stabilized frames  $O_i$ . We need to warp  $I_i$  to  $O_i$  using the correspondences of all pixels. A simple consideration is to copy the value of pixel  $\mathbf{x}$  in  $I_i$  to the position  $S_i(\mathbf{x})$  in  $O_i$ . But unfortunately,  $S_i(\mathbf{x})$  is the result of Gaussian smoothing, which is usually not integer, so it does not correspond to a pixel directly. We tried to use the rounded value of  $S_i(\mathbf{x})$  so that we can copy the pixel value directly, but from Fig. 2a, we can see that many pixels cannot be assigned a value because of the discontinuity of using rounded values. Instead, we use a weighted average algorithm to calculate the pixel value for every pixel  $\mathbf{x}'$  in  $O_i$ . In Fig. 2b, we can see that most of these pixels are assigned a value using our algorithm.

In an ordinary warping process (such as affine or perspective warping), the corresponding position  $S_i^{-1}(\mathbf{x}')$  (not integer) in input frame  $I_i$  for every pixel  $\mathbf{x}'$  (integer) in output frame  $O_i$  is known, which is the inverse situation to our problem. Bilinear interpolation is generally used in this situation to get the value for every pixel in output frame using four nearest pixels in the input frame. Inspired by bilinear interpolation, we make every pixel  $\mathbf{x}$  in  $I_i$  contribute to four pixels in  $O_i$  nearest to  $S_i(\mathbf{x})$ . Every pixel  $\mathbf{x}'$  in  $O_i$  may be affected by more than one pixels in  $I_i$ , so a weighted average value is calculated. The weight in our algorithm is the same as in bilinear interpolation. The algorithm detail of this step is shown in Algorithm 1. Note that the second output  $S_i^{-1}(\mathbf{x}')$  is used in the image completion step (Section 3.5).

---

#### Algorithm 1 Weighted average frame warping

---

**Input:** Value  $V_i^I(\mathbf{x})$  of every pixel  $\mathbf{x}$  in input frame  $I_i$ ,  
 Stabilized position  $S_i(\mathbf{x})$  in  $O_i$  of every pixel  $\mathbf{x}$  in  $I_i$ .  
**Output:** Value  $V_i^O(\mathbf{x}')$  of every pixel  $\mathbf{x}'$  in  $O_i$ ,  
 Original position  $S_i^{-1}(\mathbf{x}')$  of every pixel  $\mathbf{x}'$  in  $O_i$ .

- 1: **for all** pixel  $\mathbf{x}'$  in  $O_i$  **do**
- 2:      $V_i^O(\mathbf{x}') = 0$
- 3:      $w(\mathbf{x}') = 0$
- 4:      $S_i^{-1}(\mathbf{x}') = 0$
- 5: **end for**
- 6: **for all** pixel  $\mathbf{x} = (x, y)$  in  $I_i$  **do**
- 7:      $nearest\_pixels =$   
        $\{(\lfloor x' \rfloor, \lfloor y' \rfloor), (\lfloor x' \rfloor, \lceil y' \rceil), (\lceil x' \rceil, \lfloor y' \rfloor), (\lceil x' \rceil, \lceil y' \rceil)\}$
- 8:     **for all**  $\mathbf{tx} = (tx, ty)$  in  $nearest\_pixels$  **do**
- 9:          $weight = (1 - abs(tx - x))(1 - abs(ty - y))$
- 10:          $V_i^O(\mathbf{tx}) = V_i^O(\mathbf{tx}) + V_i^I(\mathbf{x}) \cdot weight$
- 11:          $w(\mathbf{tx}) = w(\mathbf{tx}) + weight$
- 12:          $S_i^{-1}(\mathbf{tx}) = S_i^{-1}(\mathbf{tx}) + \mathbf{x} \cdot weight$
- 13:     **end for**
- 14: **end for**
- 15: **for all** pixel  $\mathbf{x}'$  in  $O_i$  **do**
- 16:      $V_i^O(\mathbf{x}') = V_i^O(\mathbf{x}') / w(\mathbf{x}')$
- 17:      $S_i^{-1}(\mathbf{x}') = S_i^{-1}(\mathbf{x}') / w(\mathbf{x}')$
- 18: **end for**

---

### 3.5 Image completion

Some pixels in the output frames may still not be assigned a value after the warping step. From Algorithm 1 we can see that our warping method is applied on every pixel in the input frames rather than directly on pixels in output frames. Every pixel in input frames affects four nearest pixels in output frames but it is not ensured that all pixels in output frames



**Fig. 2** Performance of weighted average algorithm in frame warping. **a** Result using rounded values. Many pixels remain empty. **b** Result using weighted average values. Most pixels are filled

can be affected. Fortunately, the number of these empty pixels is usually very small. In our experiments, the number of empty pixels is usually less than 1 % of the frame pixel number.

In order to generate complete output frames, an image completion step is proposed. Our image completion algorithm is also based on PatchMatch. The PatchMatch method [1] is originally designed for structural image editing, including image completion. In fact, we can directly use PatchMatch or any other image completion algorithm to fill the empty pixels. But since we have already known the correspondences of most pixels between each pair of input and output frame, we can simplify this procedure using these information.

In Algorithm 1, the second output is calculated for assisting image completion.  $S_i^{-1}(\mathbf{x}')$  is approximate original position of every pixel  $\mathbf{x}'$  in  $O_i$  if it is not empty. This is not the exact value but it can narrow down the searching range for empty pixels nearby. Firstly we find out all empty pixels. We identify empty pixels by setting  $empty(\mathbf{x}) = true$  according the output of Algorithm 1:

$$empty(\mathbf{x}) = \begin{cases} True, & S_i^{-1}(\mathbf{x}') = 0 \\ False, & Otherwise \end{cases} \tag{8}$$

Then we find the corresponding patch in  $I_i$  for every empty pixel (in fact, the patch centering at this pixel) in  $O_i$  with least distance and copy the center pixel’s value to fill the empty pixel. In order to find the patch with least distance, we simply apply PatchMatch between  $O_i$  and  $I_i$ , but we add following limitation to make it work more efficiently:

1. The values of non-empty pixels should not be modified, i.e. the PatchMatch algorithm are applied only on empty pixels. This greatly reduces computation.
2. We use  $S_i^{-1}(\mathbf{x})$  as initial value for every pixel  $\mathbf{x}$  in  $O_i$  rather than a random value. Values of non-empty pixels has been calculated in Algorithm 1. For every empty pixel  $\mathbf{x}$ , it is initiate as average of values of neighbour non-empty pixels:

$$S_i^{-1}(\mathbf{x}) = \frac{\sum_{\mathbf{y} \in N_{ne}(\mathbf{x})} (S_i^{-1}(\mathbf{y}))}{|N_{ne}(\mathbf{x})|} \tag{9}$$

$N_{ne}(\mathbf{x}) = \{\mathbf{y} || \mathbf{y} - \mathbf{x} || < th, empty(\mathbf{y}) = false\}$  and  $|N_{ne}(\mathbf{x})|$  is the element number of  $N_{ne}$ .  $th$  is set to 5 in our experiments.

3. The window of random search is restricted to a window centering at the initial position  $S_i^{-1}(\mathbf{x})$ . Because of the spacial continuity of video frames, the correspondence of every empty pixel ought to be near neighbour pixels so we do not have to search patches far away. The window of random search is set to  $30 \times 30$  in our experiments.
4. Note that the patch centering at one empty pixel may contain other empty pixels. These pixels are ignored when calculating patch distance.

Figure 3 presents some frames before and after our image completion step.



**Fig. 3** Performance of image completion algorithm. *Top*: Frames before image completion. Some pixels are empty (black pixels in *red circles*). *Bottom*: Frames after image completion. All pixels get appropriate values

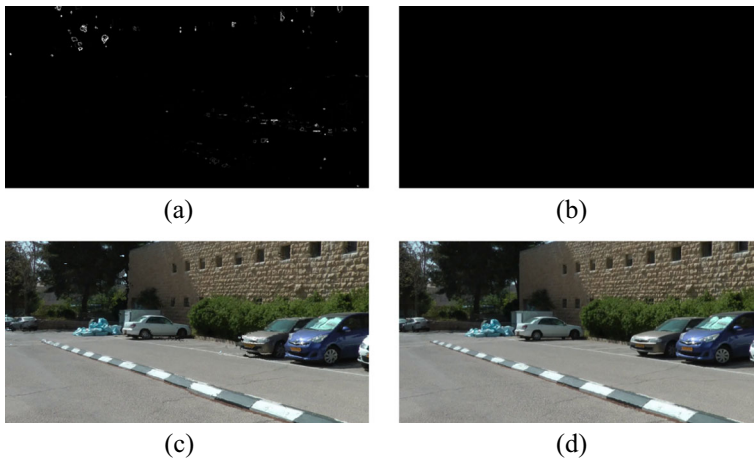
## 4 Results and comparisons

We have tested our algorithm on many different videos, which cover a variety range of scenes, such as indoor and outdoor scenes. All videos used in our experiment are captured by hand-held cameras and there is obvious jitter in all videos. Many successful results show that our method is effective and robust. In Fig. 4, we show the frames of some results. Since videos can not be shown in this paper, please refer to accompanying videos. We also compare our method with some previous methods. We discuss the comparison in this section.

**Comparison with PatchMatch** The algorithm of motion estimation step in our method is based on PatchMatch. PatchMatch can find nearest neighbour patches very quickly. Although it takes spacial coherency into consideration, it does not force this constraint. However, in our experiment, we find that spacial coherency is very important in stabilization. So we force this constraint in our motion estimation step. Figure 5 shows the comparison of our method and PatchMatch on one frame. Figure 5a and b shows the



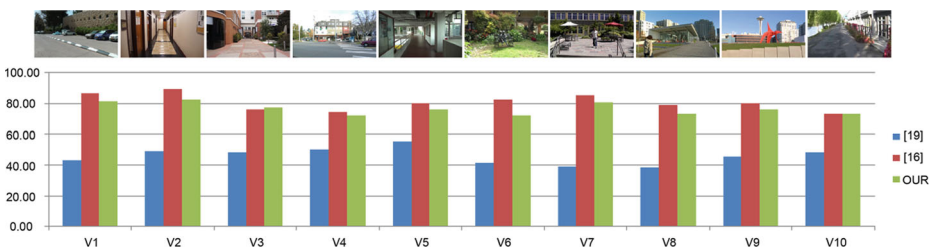
**Fig. 4** Some results of our method



**Fig. 5** Comparison with PatchMatch. **a** Spatial coherency energy of result of PatchMatch. **b** Spatial coherency energy of result of our motion estimation algorithm. **c** Final stabilization result after (a). **d** Final stabilization result after (b)

spatial coherency energy (the second energy term in (3)) of the matching results of PatchMatch and our algorithm respectively. Whiter pixel indicates larger energy at that position. We can obviously see that the spatial coherency energy is much smaller in our algorithm. The final stabilization results using these two algorithms are shown in Fig. 5c and d and our result is also much better.

**User study** A user study is conducted to compare the performance of our method with other methods. Two previous methods are used in this user study: one of them [19] uses single homography warping, and the other one [16] uses multiple homographies warping. 10 sets of videos are used and each video set consists of an original shaky video and 3 stabilized results (including results of our method and the two methods above). Each set of videos is displayed to the 78 participants and they are asked to give a score between 0 and 100 for every stabilization result. Better stabilization performance gets bigger score. The results are displayed randomly and anonymously to the participants. The snapshots of ten shaky videos and the average scores of the 10 sets of videos are shown in Fig. 6. Some detailed information of the 10 shaky videos is shown in Table 1.



**Fig. 6** User study. *Top*: Snapshots of ten shaky videos. *Bottom*: Scores of ten sets of stabilized videos using [16, 19] and our method

**Table 1** Information of shaky videos used in user study

	Duration	Frame rate	Format	Resolution
v1	14s	30	avi	800 × 450
v2	8s	29	mp4	640 × 360
v3	12s	29	mp4	640 × 360
v4	16s	29	mp4	1280 × 720
v5	17s	50	avi	640 × 360
v6	12s	30	avi	640 × 360
v7	8s	29	mp4	1280 × 720
v8	15s	29	mp4	1280 × 720
v9	9s	30	avi	640 × 360
v10	9s	30	avi	640 × 360

From Fig. 6 we can see that [19] gets the lowest scores in all cases and the scores of [16] are obviously higher than [19]. The scores of our method are very close to those of [16]. Although the scores of our method are a little lower in most cases, the difference of visual effect is too small to be perceived. Based on our experience, the difference in stability of two videos can hardly be distinguished by human eyes if the difference in scores is less than 10. In the user study, participants are asked to give an accurate score to every video, so they may watch every video for many times carefully before giving the scores. In practice, when the videos are watched normally for only once, their performances are quite comparable. This can be easily understood by taking a glance at the demo we proposed together with this paper. The comparisons between our method and these two categories of methods are shown below.

**Comparison with methods using single homography warping** Single homography warping is used widely in early methods, especially in 2D methods, such as [5, 19]. These methods are based on single plane assumption, so they cannot handle videos with complex scene well. Different from these methods, our method directly handle every pixel separately no matter whether they are on the same plane or not. From Fig. 6 we can see that our method gets much higher scores than [19].

**Comparison with methods using multiple homographies warping** Multiple homographies warping is proposed in order to avoid the disadvantage of single homography warping. Different homographies are used in different area in each frame. This kind of method includes [13, 16] and [26]. A step further, our method calculate the stabilized position for every pixel in the frames, which generates comparable results to state-of-art methods. From Fig. 6 we can see that the scores of our methods are very close to those of [16].

**Quantitative evaluation** It is a difficult problem to evaluate the stability of a video using quantitative metric. In most previous methods, there are only subjective evaluation or comparisons. In [16], three metrics are proposed to evaluate the quality and stability of result videos. However, the stability metric of [16] require the bundled camera paths calculated in this method, which is not suitable for our method. We propose another metric that is suitable for other methods. Between each pair of adjacent frames, the correspondence of every pixel is already known. The distance of correspondence pixels is the motion of that pixel.

**Table 2** Motion in videos before and after stabilization

	Origin	[19]	[16]	Our
v1	4.8	2.3	1.3	1.4
v2	3.5	2.0	1.3	1.3
v3	7.6	3.2	1.2	1.3
v4	9.8	4.0	1.6	1.5
v5	4.4	2.5	2.0	2.1
v6	1.0	0.9	0.8	0.8
v7	5.8	1.7	1.0	0.9
v8	4.3	1.9	1.1	1.0
v9	8.9	3.2	1.9	2.0
v10	5.5	2.7	1.7	1.6

We use the average motion of all pixels in a video to represent the motion of the video. The motion data of original videos and result videos of our method and [16, 19] are shown in Table 2. We can see that the motion of shaky videos is greatly reduced using our method.

**Limitation** The major limitation of our method is the low computation speed. Video processing usually costs much time since a video contains many frames. Although all the four steps of our method do not cost too much time, the total time cost to stabilize a video is quite large. Another limitation of our method is that it does not works well on moving objects because the motion of moving objects is not regular and is difficult to estimate.

**Future work** One future work is to speed up this method. The high precision of our method slows down the computation speed. Maybe we do not need to handle every pixel so that the computation burden can be reduced. Besides, we need to improve the performance on moving objects. One consideration is to pick out move objects from video and handle them in a different way. So far, there is not a widely accepted numerical standard to evaluate the stability of videos. Therefore, the evaluation of results and comparison between different methods are accomplished by human eyes at most time. In order to avoid subjective evaluation, a numerical stability evaluation method is strongly needed.

## 5 Conclusions

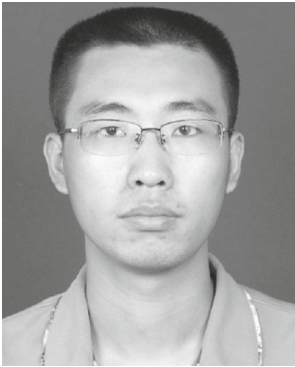
In this paper, we present a novel pixel-wise video stabilization method. Instead of using model based on sparse feature points, dense motion field is used to model the motion between adjacent frames. A patch based motion estimation algorithm is proposed to get the dense motion field, followed by an Gaussian smoothing step. Frames are warped using an weighted average warping algorithm and empty pixels are filled using a patch based image completion algorithm. Our method can handle videos with complex scenes well. Many successful results show the effectiveness of our method.

**Acknowledgments** We would like to thank the anonymous reviewers for their helpful advices. We also thank the participants in our user study. This work is supported by the National Natural Science Foundation of China under Project No. 61133008 and No. 61202180.

## References

1. Barnes C, Shechtman E, Finkelstein A, Goldman D (2009) Patchmatch: a randomized correspondence algorithm for structural image editing. In: *ACM Transactions on Graphics-TOG*, vol 28. doi:[10.1145/1531326.1531330](https://doi.org/10.1145/1531326.1531330)
2. Chen Z, Jin H, Lin Z, Cohen S, Wu Y (2013) Large displacement optical flow from nearest neighbor fields. In: *2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp 2443–2450. doi:[10.1109/CVPR.2013.316](https://doi.org/10.1109/CVPR.2013.316)
3. Chouvatut V, Madarasmis S (2013) Tuceryan, M., 3d face and motion estimation from sparse points using adaptive bracketed minimization. In: *Multimedia tools and applications*, vol 63, p 569589. doi:[10.1007/s11042-011-0925-8](https://doi.org/10.1007/s11042-011-0925-8)
4. Furht B (1996) *Motion Estimation Algorithms for Video Compression*. Springer, Westwater R
5. Gleicher ML, Liu F (2008) Re-cinematography: Improving the camera dynamics of casual video. In: *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol 5. doi:[10.1145/1404880.1404882](https://doi.org/10.1145/1404880.1404882)
6. Goldstein A, Fattal R (2012) Video stabilization using epipolar geometry. In: *ACM Transactions on Graphics(TOG)*, vol 31. doi:[10.1145/2231816.2231824](https://doi.org/10.1145/2231816.2231824)
7. Grundmann M, Kwatra V, Essa I (2011) Auto-directed video stabilization with robust l1 optimal camera paths. In: *2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, p 225232. doi:[10.1109/CVPR.2011.5995525](https://doi.org/10.1109/CVPR.2011.5995525)
8. He K, Sun J (2012) Computing nearest-neighbor fields via propagation-assisted kd-trees. In: *2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, p 111118. doi:[10.1109/CVPR.2012.6247665](https://doi.org/10.1109/CVPR.2012.6247665)
9. Irani M, Rousso B, Peleg S (1994) Recovery of ego-motion using image stabilization. In: *1994 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, p 454460. doi:[10.1109/CVPR.1994.323866](https://doi.org/10.1109/CVPR.1994.323866)
10. Korman S, Avidan S (2011) Coherency sensitive hashing. In: *2011 IEEE International Conference on Computer Vision (ICCV)*, pp 1607–1614. doi:[10.1109/ICCV.2011.6126421](https://doi.org/10.1109/ICCV.2011.6126421)
11. Lee KY, Chuang YY, Chen BY, Ouhyoung M (2009) Video stabilization using robust feature trajectories. In: *2009 IEEE 12th International Conference on Computer Vision*, p 13971404. doi:[10.1109/ICCV.2009.5459297](https://doi.org/10.1109/ICCV.2009.5459297)
12. Litvin A, Konrad J, Karl WC (2003) Probabilistic video stabilization using Kalman filtering and mosaicing. In: *Symposium on Electronic Imaging, Image and Video Communications*, p 663674. doi:[10.1117/12.476436](https://doi.org/10.1117/12.476436)
13. Liu F, Gleicher M, Jin H, Agarwala A (2009) Content-preserving warps for 3D video stabilization. In: *ACM Trans. Graphic*, p 28. doi:[10.1145/1531326.1531350](https://doi.org/10.1145/1531326.1531350)
14. Liu F, Gleicher M, Wang J, Jin H, Agarwala A (2011) Subspace video stabilization. In: *ACM Trans. Graphic*, vol 30. doi:[10.1145/1899404.1899408](https://doi.org/10.1145/1899404.1899408)
15. Liu S, Wang Y, Yuan L, Bu J, Tan P, Sun J (2012) Video stabilization with a depth camera. In: *Proceedings of the CVPR*, p 8995. doi:[10.1109/CVPR.2012.6247662](https://doi.org/10.1109/CVPR.2012.6247662)
16. Liu S, Yuan L, Tan P, Sun J (2013) Bundled camera paths for video stabilization. In: *ACM Transactions on Graphics (TOG)*, p 32. doi:[10.1145/2461912.2461995](https://doi.org/10.1145/2461912.2461995)
17. Lowe DG (1999) Object recognition from local scale-invariant features. In: *The proceedings of the seventh IEEE international conference on Computer vision*, vol 2, pp 1150–1157. doi:[10.1109/ICCV.1999.790410](https://doi.org/10.1109/ICCV.1999.790410)
18. Lu J, Yang H, Min D, Do MN (2013) Patch match filter: Efficient edge-aware filtering meets randomized search for fast correspondence field estimation. In: *2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp 1854–1861. doi:[10.1109/CVPR.2013.242](https://doi.org/10.1109/CVPR.2013.242)
19. Matsushita Y, Ofek E, Tang X, Shum HY (2005) Full-frame video stabilization. In: *Proceedings of the CVPR*, p 5057. doi:[10.1109/CVPR.2005.166](https://doi.org/10.1109/CVPR.2005.166)
20. Morimoto C, Chellappa R (1996) Automatic digital image stabilization. In: *Proceedings of the CVPR*
21. Morimoto C, Chellappa R (1998) Evaluation of image stabilization algorithms. In: *1998 IEEE International Conference on Acoustics, Speech and Signal Processing*, vol 5, p 27892792. doi:[10.1109/ICASSP.1998.678102](https://doi.org/10.1109/ICASSP.1998.678102)
22. Okade M, Biswas PK (2014) Video stabilization using maximally stable extremal region features. In: *Multimedia Tools and Applications*, vol 68, p 947968. doi:[10.1007/s11042-012-1095-z](https://doi.org/10.1007/s11042-012-1095-z)
23. Shi J, Tomasi C (1994) Good features to track. In: *1994 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp 593–600. doi:[10.1109/CVPR.1994.323794](https://doi.org/10.1109/CVPR.1994.323794)

24. Shuaicheng LL, Yuan PT, Sun J (2014) Steadyflow: Spatially smooth optical flow for video stabilization. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 4209–4216. doi:[10.1109/CVPR.2014.536](https://doi.org/10.1109/CVPR.2014.536)
25. Walha A, Wali A, Alimi AM (2014) Video stabilization with moving object detecting and tracking for aerial video surveillance. In: Multimedia Tools and Applications, p 123. doi:[10.1007/s11042-014-1928-z](https://doi.org/10.1007/s11042-014-1928-z)
26. Wang Y, Liu F, Hsu P, Lee T (2013) Spatially and temporally optimized video stabilization. In: IEEE Transactions on Visualization and Computer Graphics, pp 1354–1361. doi:[10.1109/TVCG.2013.11](https://doi.org/10.1109/TVCG.2013.11)
27. Wang ZQ, Zhang L, Huang H (2013) Multiplane video stabilization. In: Computer Graphics Forum, vol 32, pp 265–273. doi:[10.1111/cgf.12234](https://doi.org/10.1111/cgf.12234)
28. Zhang G, Hua W, Qin X, Shao Y, Bao H (2009) Video stabilization based on a 3D perspective camera model. In: The Visual Computer, vol 25, p 9971008. doi:[10.1007/s00371-009-0310-z](https://doi.org/10.1007/s00371-009-0310-z)
29. Zhou Z, Jin H, Ma Y (2013) Plane-based content-preserving warps for video stabilization. In: 2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 2299–2306. doi:[10.1109/CVPR.2013.298](https://doi.org/10.1109/CVPR.2013.298)



**Zhongqiang Wang** was born in 1988. He received the B.S. degree from Xi'an Jiaotong University Xian, China, in 2010. He is a doctor candidate in school of Electronic and Information Engineering, Xi'an Jiaotong University. His current research interests include image and video processing.



**Hua Huang** was born in 1975. He received the B.S., M.S., and Ph.D. degrees from Xi'an Jiaotong University, Xian, China, in 1996, 2001, and 2006, respectively. He is currently a Professor in School of Computer, Beijing Institute of Technology. His current research interests include image and video processing, computer graphics, and pattern recognition.